



ARL-SR-0312 • MAR 2015



US Army Research Laboratory

Adaptive Geometry Shader Tessellation for Massive Geometry Display

by Lee A Butler and Clifford Yapp

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.



Adaptive Geometry Shader Tessellation for Massive Geometry Display

by Lee A Butler and Clifford Yapp
Survivability/Lethality Analysis Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY)		2. REPORT TYPE		3. DATES COVERED (From - To)	
March 2015		Final		1 October 2013–30 September 2014	
4. TITLE AND SUBTITLE Adaptive Geometry Shader Tessellation for Massive Geometry Display				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Lee A Butler and Clifford Yapp				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-SLB-S Aberdeen Proving Ground, MD 21005-5068				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-SR-0312	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT A large number of the Survivability/Lethality Analysis Directorate's (SLAD's) software tools need to display geometric data. This project explored options for improving the speed and clarity with which SLAD's tools can display geometry. The goal was to reduce the amount of human effort and elapsed time necessary to prepare complex models for use in analysis and visualization tasks. We investigated several avenues for high-speed visualization and worked to update BRL-CAD's graphics display system to support more modern display layers. While additional work remains, we identified high-performance techniques and achieved the first stages of display system improvements.					
15. SUBJECT TERMS temporal coherence, dynamic occlusion culling, occlusion queries, geometry, visualization					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code)
Unclassified	Unclassified	Unclassified	UU	22	Clifford Yapp (410) 278-1382

Contents

List of Figures	iv
Executive Summary	v
1. Introduction and Background	1
2. Approach	2
3. Speed Improvements in the Visual Simulation Laboratory	2
4. Ray Tracing	4
5. Sharing Display Technologies Across Applications	5
6. Conclusions and Future Work	7
7. References	8
Appendix. Refactoring and Updating Steps for Improving BRL-CAD's Display System	9
List of Symbols, Abbreviations, and Acronyms	13
Distribution List	14

List of Figures

Fig. 1	Example frame timing code	3
Fig. 2	The expectation was that the time required for the interFrame process would remain constant. Above 400 objects there is a consistent increase of interframe process overhead.	4
Fig. 3	Drawing of MGED wireframe and ray-traced image in OpenSceneGraph-provided OpenGL context	6
Fig. 4	Working portable FontSplash text display in OpenSceneGraph-provided OpenGL context.....	6

Executive Summary

Survivability/Lethality Analysis Directorate (SLAD) customers would like a faster product that is easier to understand and use. To address this desire, we conducted work under a SLAD Methodology Research Initiative to explore options for improving the speed and clarity with which SLAD's tools can display geometry. The goal was to reduce the amount of human effort and elapsed time necessary to prepare complex models for use in analysis and visualization tasks.

To investigate the possibility of fast geometry visualization within SLAD's tools, we focused on 2 task-oriented goals: 1) investigate the OpenSceneGraph (OSG) 3-dimensional graphics toolkit's¹ existing capabilities and any other available avenues for speed and 2) redesign and update BRL-CAD's graphics display system to support OSG.

After investigating the existing capabilities provide by OSG, we found that enabling the Coherent Hierarchical Culling (CHC) algorithm² resulted in successful renderings of complex computer-aided design geometry at an excellent 140 frames/s, but the required memory (24 GB for a 2-GB geometry model) and preparation time (hours on computer hardware circa 2014) are currently prohibitive for practical use. Further work might improve our implementation of the CHC algorithm, but concurrent experiments with another approach using Intel's Embree raytracing engine indicate raytracing requires approximately 30 s of preparation time to achieve a rendering speed of 15–30 frames/s, which is adequate for most visualization purposes when combined with a traditional rasterized rendering system for interactive elements. Therefore, we must explore the possibilities of the Embree engine in the context of analytical raytracing.

An additional problem arose when performance testing indicated that a significant portion of the bottleneck in OSG's rasterization-based rendering occurs outside the drawing phase. Further investigation is being pursued under separate funding.

Redesign and update of BRL-CAD's display system proved to be an involved problem touching a large portion of BRL-CAD's core code. Modernization efforts have improved BRL-CAD's code to a point where OSG can be used to provide an OpenGL drawing canvas portably, but it is not yet to the point where BRL-CAD can leverage the higher-level OSG functionality. OSG has been incorporated into

¹ OpenSceneGraph web site. Open Source High Performance 3D Graphics Toolkit [accessed 2015 Feb 10]. <http://www.openscenegraph.org>.

² Mattausch O, Bittner J, Wimmer M. CHC++: coherent hierarchical culling revisited. Computer Graphics Forum (Proceedings Eurographics 2008). 2008 Apr;27(2):221–230. Also available at <http://www.cg.tuwien.ac.at/research/publications/2008/mattausch-2008-CHC/>.

the BRL-CAD build and deployment infrastructure. Substantial additional refactoring of application and library code is necessary to fully leverage high-performance rendering. For BRL-CAD to remain a part of the US Army Research Laboratory's (ARL's) core software infrastructure, such improvements are essential, for they provide long-term benefits in terms of both maintenance cost reductions and enabling desired improvements across ARL's modeling and simulation tools.

1. Introduction and Background

Survivability/Lethality Analysis Directorate (SLAD) customers would like a faster product that is easier to understand and use. To address this desire, we conducted work under a SLAD Methodology Research Initiative (MRI) to explore options for improving the speed and clarity with which SLAD's tools can display geometry. The goal was to reduce the amount of human effort and elapsed time necessary to prepare complex models for use in analysis and visualization tasks.

The US Army Research Laboratory's (ARL's) vulnerability/lethality (V/L) analysis toolbox has 2 significant display systems for 3-dimensional (3-D) model information. The oldest is descended from Mike Muuss's original graphics editor work in the 1980s¹ and now forms the graphical display system of the BRL-CAD open-source computer-aided design package. The other is developed for the Visual Simulation Laboratory (VSL) built atop the OpenSceneGraph (OSG) 3-D graphics toolkit.²

The original work to create the current BRL-CAD display manager (vector list drawing) and frame buffer (pixel drawing) interfaces took place circa 1982, and relatively little has changed since that time. Both interfaces were created as common Application Programming Interfaces (APIs) for special purpose hardware devices of that era. As such, they represented "lowest common denominator" functionality. Interfaces for modern display mechanisms such as OpenGL have been implemented for these APIs, but to date, the level of functionality expressed in the API has not been expanded to take advantage of new capabilities.

Although BRL-CAD's libraries fulfilled their original purpose, in today's graphics environment this approach represents a significant obstacle to adopting the current industry-standard practices for displaying this type of data. For example, in modern systems, such as OpenGL and DirectX, there is no distinction between pixel, vector, and polygon drawing hardware or associated APIs. This indicates that management of drawing information and scenes within BRL-CAD should be reworked to support leveraging modern features of graphics hardware and software APIs.

Most of ARL's specialized V/L applications require 3-D geometry display. They use the BRL-CAD software libraries for these functions. This propagates the limitations and technical debt of BRL-CAD through the tool chain.

VSL is a newer project, deliberately setting out to employ improvements in geometric display research and technology from the last 20 years to addressing

V/L-related application needs. VSL is able to use much of what OSG offers by avoiding the low-level approach taken by BRL-CAD's system, which allows for more sophisticated, higher-performance visualizations. However, this updated design also means that the resulting features cannot be easily integrated into other analysis tools at ARL that generally expect a more dated approach to drawing. Recent experience has shown that even OSG is not adequate for some large target models provided by manufacturers for V/L analysis.

The problems addressed by this research are 1) what approaches might be taken to improve performance when using a modern graphics display system and 2) what must be done to update the older BRL-CAD display system to let it take advantage of modern improvements.

2. Approach

Given the goals and the software resources available, the approach separated cleanly into 2 task-oriented goals: 1) investigate OSG's existing capabilities and any other available avenues for speed improvements that could be integrated into the OSG framework in place within VSL and 2) redesign and update BRL-CAD's graphics display system to support OSG.

3. Speed Improvements in the Visual Simulation Laboratory

Numerous techniques for speeding up the drawing process exist. Frustum culling discards geometry that is clearly outside the field of view of the virtual camera prior to rendering. Back-face culling discards geometry that is facing away from the camera. These are both provided in robust and effective means by the most current scene management systems. This work focused on occlusion culling, which attempts to discard geometry that cannot be seen because other geometry is between it and the camera position.

Occlusion culling requires some knowledge of which objects are visible and which are not. Even in changing scenes, objects in view (or not) in one frame are likely to be in view (or not) in a subsequent frame. Rasterization engines, such as OpenGL and DirectX, allow for a limited number of queries about whether objects are occluded. Occlusion queries are relatively lightweight instructions that return the number of visible pixels of geometry (or a simplified proxy shape) without the need of reading back the frame buffer. The technique consists of several stages:

- 1) A query number is obtained from the graphics driver.

- 2) The query state is entered for the obtained query number.
- 3) Drawing operations are performed.
- 4) The query state is exited for the query number.
- 5) The driver is asked if the results of the query number are available.
- 6) When results are available, the host CPU gets the results.
- 7) Typically, objects that are not visible are not drawn in subsequent frames, or are drawn as a simpler tessellation (even an axis-aligned bounding box). The simpler geometry may not be rendered to the color buffer but only depth-checked. This drastically reduces the drawing necessary to complete the scene.
- 8) Historically, the queries were conducted per frame. In more recent work, such as that of Mattausch et al.,³ a hierarchical approach to querying is used, and sets of queries are aggregated to reduce overhead.

The test geometry used for this work was from a modern military system being subjected to ballistic analysis. It was chosen because it is representative of the detail and complexity in use and stresses the existing display capability in VSL. The geometry represented about 2 GB of polygonal data.

In timing testing (Fig. 1) with the example code provided by Mattausch et al.,³ we found that the depth of the bounding volume hierarchy (BVH) had to be increased substantially compared to the author's use to see any significant speed. Unfortunately, this led to excessively long run-times for BVH construction. We anticipated that this cost could be paid once as a preprocessing step for completed target geometry and the BVH used for all future display needs. Eventually, the performance levels indicated in the literature were achieved (~140 frames/s). However, the BVH data needed to achieve these levels consumed approximately 24 GB of memory. This was most of the available system memory on a typical high-end workstation.

```
DoFrame () {  
    interFrame = timer.elapsed();  
  
    timer.start();  
    draw();  
    drawTime = timer.elapsed();  
  
    requestRedrawLater(10ms);  
    timer.start();  
}
```

Fig. 1 Example frame timing code

While the work testing Mattausch’s example code was ongoing, we instrumented the traditional drawing infrastructure in the VSL framework. This revealed an unexpected source of delay. Experiments with progressively increasing geometry complexity showed that the drawing time (drawTime) increased relatively slowly and linearly. The time between drawing frames (interFrame) did not remain constant as expected (Fig. 2). Further investigation to identify the cause of this delay (and eliminate it) is being pursued under VSL project funding.

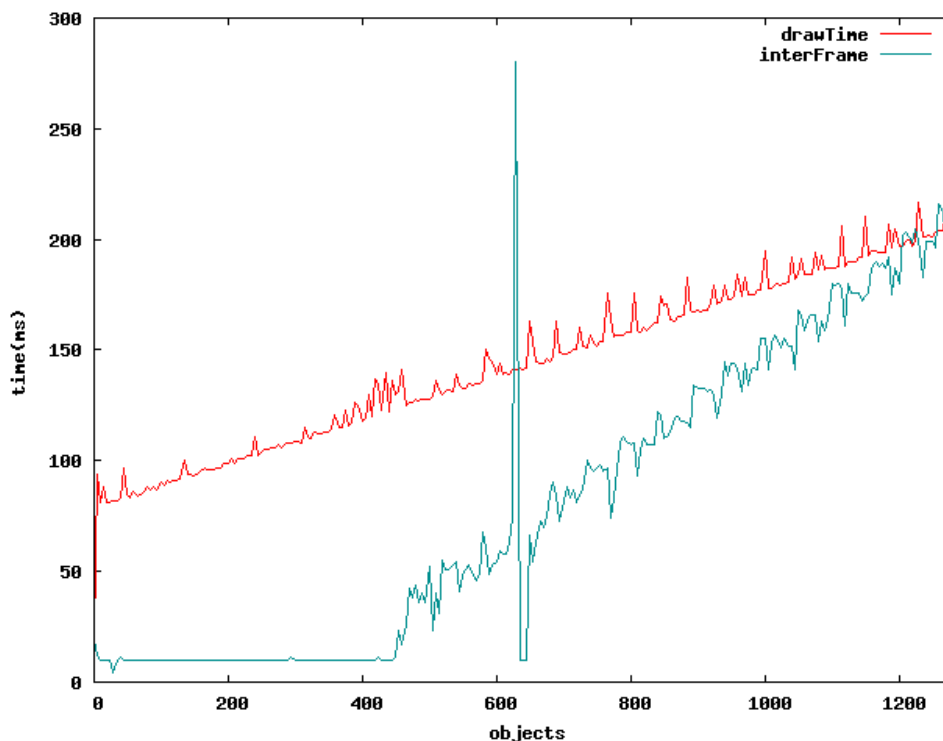


Fig. 2 The expectation was that the time required for the interframe process would remain constant. Above 400 objects there is a consistent increase of interframe process overhead.

4. Ray Tracing

While this work was ongoing, a colleague working on a separate project was investigating modern ray-tracing engines. We loaded the test geometry into a prototype renderer using the Intel Embree ray tracer. We were pleasantly surprised to see that the BVH for the test geometry was constructed in approximately 30 s and rendering was possible at a satisfying 30 frames/s. This strongly suggests that a hybrid raytracing/rasterization rendering system would be ideal for SLAD applications. Both NVIDIA and Intel have created such systems. Since SLAD applications often require a high-performance ray tracer, it is a minor extra cost to leverage the ray tracer to assist rendering.

5. Sharing Display Technologies Across Applications

The first order of business for improving BRL-CAD’s graphics display system was developing an understanding of the existing code. BRL-CAD-based programs, such as the Multiple Device Geometry Editor (MGED) (and the newer “libged” library), have significant amounts of code that interleaves application functionality with drawing operations. This lack of modularity makes integration of modern practices and optimizations particularly challenging.

While examining the existing display architecture and some initial design work to decide what an “ideal” libdm API would look like, we found that the only practical approach to the complexities of reworking the existing code base required a series of incremental refactoring steps (see the Appendix).

While it was not possible to complete all necessary work in FY14, OSG was integrated into BRL-CAD’s build system, and enough of the refactoring process was completed to make an OSG-managed OpenGL context practical as a drawing canvas for BRL-CAD’s system. Figure 3 demonstrates a traditional BRL-CAD MGED ray-traced image in combination with an overlaid wireframe drawn on an OSG-provided OpenGL canvas. The addition of the small FontStash⁴ library for OpenGL text drawing in combination with the portable OSG context management API resulted in the first complete portable cross-platform display manager back end for OpenGL: `osgl`. The problem of portable text drawing, in particular, has long been a problem for BRL-CAD’s display manager code, and this work is (to the best of the author’s knowledge) the first practical demonstration of a portable solution within BRL-CAD. Figure 4 demonstrates this text drawing ability in the MGED faceplate in display editing interface as well as the ellipsoid’s primitive parameter labels.

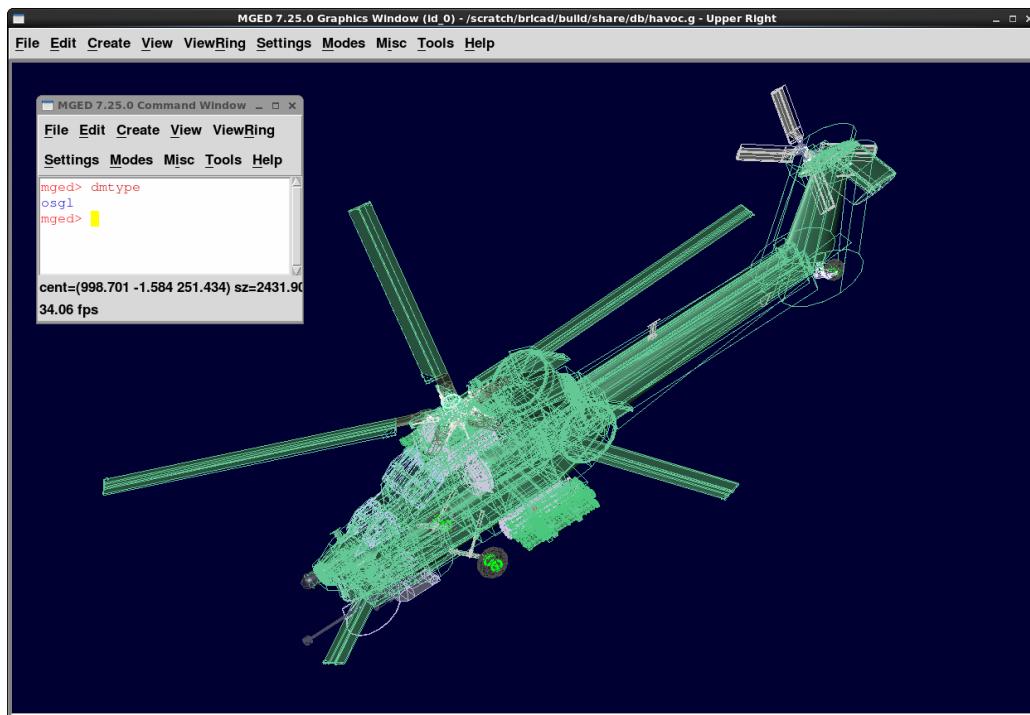


Fig. 3 Drawing of MGED wireframe and ray-traced image in OpenSceneGraph-provided OpenGL context

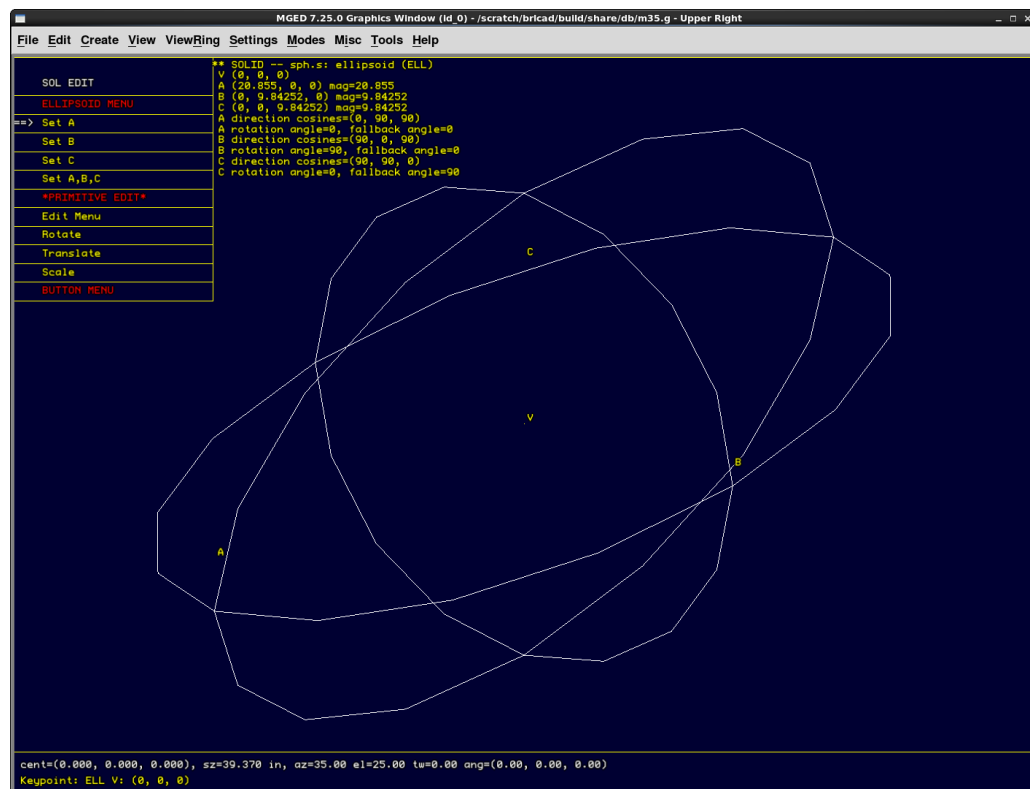


Fig. 4 Working portable FontSplash text display in OpenSceneGraph-provided OpenGL context

Once osgl has been sufficiently vetted for quality and stability, it will replace 2 platform-specific display manager back ends (the X11 back-end ogl and the Windows back-end wgl) and avoid the need for the creation of a third. (BRL-CAD does not currently have the ability to work natively on Mac OSX; this limitation was due in large part to the absence of a display manager that could work with the native Mac OSX API.) This capability alone represents a significant benefit derived from this MRI effort.

The proper introduction of scenegraph-based drawing management (instead of simply using the OSG canvas for existing drawing techniques) is a considerably more invasive change and may need to wait on other simplifications to the BRL-CAD code base (such as consolidation of the MGED and Archer interfaces) to reduce the amount of work required.

6. Conclusions and Future Work

Given that BRL-CAD forms the basis for all of Ballistics Vulnerability/Lethality Division's geometry processing capability for all of its analysis tools, it is critical to focus on maintaining and extending the capabilities to assure that BRL-CAD and derived tools remain relevant and usable in the future.

Scene graph-based drawings in OSG should be integrated into current tools, including BRL-CAD. The highest payoff in both performance and flexibility will be achieved with a hybrid raytracing and rasterization rendering engine, such as NVIDIA's SCENIX or Intel's Embree.

While a number of significant steps remain to be completed before proper support for scene graph displays in BRL-CAD's libdm applications becomes a reality, it is clearly possible and a necessary step if substantial improvements to BRL-CAD's 3-D visualization frameworks are to be achieved.

In proposed follow-on efforts, we will explore the feasibility of rendering BRL-CAD geometry directly in the Embree ray-tracing engine.

7. References

1. Muuss M, Applin KA, Suckling JR, Moss GS, Weaver EP, Stanley CA. GED: an interactive solid modeling system for vulnerability assessments. Aberdeen Proving Ground (MD): Army Ballistic Research Laboratory (US); 1983 Mar. Report No.: ARBRL-TR-024B0.
2. OpenSceneGraph web site. Open Source High Performance 3D Graphics Toolkit [accessed 2015 Feb 10]. <http://www.openscenegraph.org>.
3. Mattausch O, Bittner J, Wimmer M. CHC++: coherent hierarchical culling revisited. Computer Graphics Forum (Proceedings Eurographics 2008). 2008 Apr;27(2):221–230. Also available at <http://www.cg.tuwien.ac.at/research/publications/2008/mattausch-2008-CHC/>.
4. FontStash web site. A small, self contained library for text display in opengl [accessed 2015 Feb 10]. <https://github.com/memononen/fontstash>.

Appendix. Refactoring and Updating Steps for Improving BRL-CAD's Display System

These steps outline in detail the changes already made to the BRL-CAD codebase in support of display modernization and attempts to predict what work remains to reach a fully scene-graph-enabled system. Such predictions are contingent on assumptions. (For example, if Archer were to replace the Multiple Device Geometry Editor [MGED] in the BRL-CAD code base, certain categories of work below would no longer need to be completed.) The primary purpose of this Appendix is to provide future BRL-CAD developers with a convenient summary of the work done to date and a breakdown of expected additional necessary steps.

- 1) Remove the obsolete dg.h interface. This interface was long deprecated and had not been removed simply because of lack of time. There was no justification for spending the effort to update it to a new approach; it is time to finalize its removal. [\[DONE\]](#)
- 2) Make embedded frame buffers the responsibility of the display manager in which the frame buffer is being embedded, avoiding the need to expose implementation details in applications using embedded frame buffers. [\[DONE\]](#)
- 3) Refactor libdm and libfb macros directly accessing struct members into function calls. (A to-do item to allow checking for null structure values.) [\[DONE\]](#)
- 4) Hide the public structures of libdm and libfb, both to ensure libged remains independent of the implementation details of the display libraries and force the Application Programming Interface (API) to provide enough information for displays in a generic fashion. This impacts a large amount of code in libraries and applications. [\[DONE\]](#)
- 5) Refactor application code and (where needed) library code to remove the need for display manager back-end specific code in applications. An enhancement to libbu's vparse API proved necessary to support applications passing data around during the parsing operations, but the final result allows each dm back end to define its own specific options and provide them to the application for modification without requiring system-specific logic or headers. [\[DONE\]](#)
- 6) Refactor duplicate code (matrix handling for sure, probably others) from MGED, libdm, and libged into library APIs to ensure any changes needed to support mapping to OpenSceneGraph (OSG) are consistent. [\[IN PROGRESS\]](#)

- 7) Refactor logic common to libdm and libged but not needing types for either into shared functions in lower-level libraries (typically libbn, which already has some similar code for vlists). Where information does need to be shared between libdm and libged, create a header “bview.h” to define common view-related data types for all libraries and applications. This ensures the data is consistent between both environments without requiring linking dependencies. [\[DONE\]](#)
- 8) Consolidate functionality in libged that interacts with display lists (particularly the solid lists within display lists) to identify the scope of functionality necessary for a higher-level libdm API and to simplify eventual changes to the display list mechanism itself. [\[DONE\]](#)
- 9) Refactor MGED handling of solid lists to isolate logic. Extensive testing in this stage is particularly important to ensure no features are broken. [\[TO DO\]](#)
- 10) Integrate OSG and its dependencies into BRL-CAD’s src/other compilation system for dependency management and cross-platform compilation simplicity. [\[DONE\]](#)
- 11) Identify and test key concepts from OSG and the Visual Simulation Laboratory (VSL) that will be needed in the final scene graph aware back end. [\[IN PROGRESS\]](#)
- 12) Implement a high-level alternative API in libdm (while preserving existing API to avoid breaking code) for object drawing. At this stage, do not introduce an OSG scene graph but simply encapsulate higher-level logic from MGED/Archer/libged underneath a library API. [\[IN PROGRESS\]](#)
- 13) Refactor MGED and/or Archer/libged quaternion-based positioning code to a library API such as libbn—robust rotational interactions are a key component of most 3-dimensional viewing applications, and MGED’s solution to that problem has worked well. [\[TO DO\]](#)
- 14) Map view setup and manipulation concepts from MGED’s approach to that used by OSG. (Some of this work has been accomplished in prior VSL work.) [\[IN PROGRESS\]](#)
- 15) Migrate libged to the new high-level API. [\[TO DO\]](#)
- 16) Introduce scene graph concepts into the back-end logic for the new API functions. Ideally, these high-level functions will use back-end agnostic calls built from the lower-level API and MGED’s logic unless overridden

by a back end like OSG that wishes to take a different data management approach. [\[TO DO\]](#)

- 17) Investigate BRL-CAD's Archer interface and vulnerability/lethality-specific tools to determine what changes are necessary in those code bases to use the new API and implement them. [\[TO DO\]](#)
- 18) Incorporate any results from the speed enhancement work into the display manager back end. Depending on the findings of that work, it may be necessary to consider additional API design changes for optimal results. [\[TO DO\]](#)

List of Symbols, Abbreviations, and Acronyms

3-D	3 dimensional
API	Application Programming Interface
ARL	US Army Research Laboratory
BVH	bounding volume hierarchy
CHC	Coherent Hierarchical Culling
MGED	Multiple Device Geometry Editor
MRI	Methodology Research Initiative
OSG	OpenSceneGraph
SLAD	Survivability/Lethality Analysis Directorate
VSL	Visual Simulation Laboratory
V/L	vulnerability/lethality

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

2 DIRECTOR
(PDF) US ARMY RESEARCH LAB
RDRL CIO LL
IMAL HRA MAIL & RECORDS
MGMT

1 GOVT PRINTG OFC
(PDF) A MALHOTRA

1 DIR US ARMY EVAL CTR HQ
(HC) TEAE SV
P A THOMPSON
2202 ABERDEEN BLVD 2ND FL
APG MD 21005-5001

18 DIR USARL
(4 HC, RDRL SL
14 PDF) J BEILFUSS (HC)
P TANENBAUM (HC)
RDRL SLB
B BOWEN
RDRL SLE
R FLORES
RDRL SLB A
G MANNIX
RDRL SLB D
R GROTE
RDRL SLB E
M MAHAFFEY
RDRL SLB G
P MERGLER
RDRL SLB S
W BOWMAN
L BUTLER (1 HC, 1 PDF)
S MORRISON
M PERRY
N REED
G SAUERBORN
C YAPP (1 HC, 1 PDF)
RDRL SLB W
S SNEAD